Electronic Version
Stylesheet Version v1.1.1

Description

A Spacial Deblocking Method Using Limited Edge Differences Only to Linearly Correct Blocking Artifact

Background of Invention

[0001] This invention relates to video compression, and more particularly for spatial de-blocking methods.

[0002] Video data can greatly enhance the quality of a computing experience. The consumer use of the Internet took off once graphics was linked to earlier text-based web pages. Portable consumer devices such as cell phones and personal digital assistant (PDA's) are being equipped with small cameras to allow for capture of still or even video pictures. Efficient transmission of captured images over limited-bandwidth links requires some sort of compression of the images.

[0003] A number of video-compression techniques are known. Compression standards, such as those developed by the motion-picture-experts group (MPEG), have been widely adopted. These compression techniques are lossy techniques, since some of the picture information is discarded to increase the compression ratio. However, compression ratios of 99% or more have been achieved with minimal noticeable picture degradation.

[0004] Next-generation compression standards have been developed for transmitting video over wireless networks. The MPEG-4 standard provides a robust compression technique for transmission over wireless networks. Recovery can occur when parts of the MPEG-4 bit stream is corrupted.

[0005] These MPEG standards ultimately break the image up into small 16x16 pixel macroblocks or even smaller 8x8 pixel blocks. Each block can then be compressed more or less independently of other blocks, and movement of blocks can be described as highly compressed "motion vectors" rather than large bitmaps of pixels.

[0006] Figure 1 shows an image frame divided into rows and columns of blocks. The MPEG standard uses a divide-and-conquer technique in which the video sequence is divided into individual image frames known as video object planes (VOPs), and each frame is divided into rows and columns of macroblocks. Each macroblock is a rectangle of 16 by 16 pixels. Each macroblock can be further divided into 8x8 blocks.

[0007] Various window sizes and image resolutions can be supported by MPEG standards. For example, one common format is an image frame of 176 by 144 pixels. The image frame is divided into 18 rows of 8x8 blocks, with each row having 22 blocks each of 8x8 pixels. A total of 396 blocks are contained in each frame.

[0008] The blocks are arranged in a predetermined order, starting in the upper left with the first block (BLK #0). The second block, BLK #1, is to the right of BLK #0 in the first row, followed by blocks #2 to BLK #21 in the first row. The second row contains BLK #22 to BLK #43. The last row contains BLK #374 to BLK #395. Of course, other image sizes and formats can have the blocks in rows of various lengths, and various numbers of rows.

[0009] When an image frame is encoded, each block is encoded in order, starting with BLK #0 in the first row, and continuing with BLK #1, BLK #2 to BLK #21 in the first row, then BLK #22 to BLK #43 in the second row, and on until the last row with BLK #374 to BLK #395. The blocks are arranged in the bit stream into one or more video packets (VP) with a header.

[0010] Since each bock is compressed separately from other blocks, there can be noticeable discontinuities at block edges. For example, an highly-compressed image of a blue sky may have visible color-change bands caused by blocking artifacts. While the actual sky changes gradually from one shade of blue to another, only a few blue shades may be used to represent the sky in the compressed image. Several rows of blocks may code all pixels as a dark shade of blue while a next row codes all pixels as a lighter shade of blue. An abrupt color change may be noticeable at the edge of the row. When the color change boundary moves diagonally or crosses rows, the row crossing may cause a stair-step of jagged block edges to be visible.

[0011] MPEG decoders may use a de-blocking filter to reduce the visibility of such blocking artifacts. Filters can be applied along block edges, both vertical and horizontal edges. See for example the ISO/IEC JTC1/SC29/WG11 standard's working group document m4960.doc: section 9.1 De-Blocking Filter. Figures 2A-B show prior-art de-blocking filter applied to vertical and horizontal block edges. In Fig. 2A, a row of pixels V0, V1, ...V9 crosses a vertical boundary between two 8x8 blocks, BLK #N and BLK #N+1. Pixel V4 is the last pixel in BLK #N, while pixel V5 is the first pixel in block #N+1.

[0012] Due to compression, a noticeable color difference may appear between pixels V4 and V5. The prior-art de-blocking filter combines pixels V1, V2, V3, V4 in BLK #N using the S1 smoothing filter, pixels V5, V6, V7, V8 in BLK #N+1 using the S2 smoothing filter, and pixels V3, V4, V5, V6 that cross the boundary using the S3 smoothing filter.

[0013] In Fig. 2B, a column of pixels V0, V1, ...V9 crosses a horizontal boundary between two 8x8 blocks, BLK #N and BLK #N+22 in the next row. Pixel V4 is the last pixel in BLK #N, while pixel V5 is the first pixel in block #N+22.

[0014] Due to compression, a noticeable color difference may appear between pixels V4 and V5. The prior-art de-blocking filter combines pixels V1, V2, V3, V4 in BLK #N using the S1 smoothing filter, pixels V5, V6, V7, V8 in BLK #N+22 using the S2 smoothing filter, and pixels V3, V4, V5, V6 that cross the boundary using the S3 smoothing filter.

[0015] The smoothing filters can be re-applied for each of the 8 columns and each of the 8 rows in each 8x8 block, for all blocks in a frame. The pixel inputs V1, V2...V8 can be shifted to the right by one column, or down by one row, and the operations repeated on the new inputs.

[0016] Figure 3 is a diagram highlighting computational steps performed during a prior-art de-blocking process. The three 4-pixel groups S0, S1, S2 are input to vector multipliers 11, 14, 16. Vector multiplier 11 receives pixels V3, V4, V5, V6 that cross the horizontal or vertical boundary, and generates frequency component A0 as the inner product of the S0 pixel vector and the transposed ($[\ldots]^T$) discrete cosine transform (DCT) kernel $[2\ -5\ 5\ 2]^T$:

[0017] $A0 = ([2\ -5\ 5\ 2] * [V3\ V4\ V5\ V6]^T)/8$

[0018] Vector multiplier 14 receives pixels V1, V2, V3, V4 in BLK #N, and generates frequency component A1 as the inner product of the S1 pixel vector and the DCT kernel:

[0019] $A1 = ([2\ -5\ 5\ 2] * [V1\ V2\ V3\ V4]^T)/8$

[0020] Vector multiplier 16 receives pixels V5, V6, V7, V8 in BLK N+1 or N+22, and generates frequency component A2 as the inner product of the S2 pixel vector and the DCT kernel:

[0021] $A2 = ([2\ -5\ 5\ 2] * [V5\ V6\ V7\ V8]^T)/8$

[0022] These three inner products are frequency-domain components of the pixel arrays within the two blocks and crossing the block edge. The absolute values of frequency components A0, A1, A2 are generated by absolute-value generators 22, 24, 26, such as by dropping the sign bit. Minimum selector 10 then selects the minimum absolute value of A0, A1, A2. The original sign of S0 is applied to this selected minimum to generate A0'.

[0023] The original A0 is then subtracted from A0', the difference multiplied by 5 and integer-divided by 8 using calculator 28. The result is input to clipper 20, which clips the result to a value between 0 and DE. DE is half of the edge-pixel difference (V4-V5)/2, generated by pixel differencer 18. Thus clipper 20 limits extreme values to a range of 0 to (V4-V5)/2. The clipped difference value output by clipper 20 is D.

[0024] Applicator 30 then adds clipped difference D to edge pixel V5 to generate the filtered pixel V5. Clipped difference D is also subtracted from edge pixel V4 to generate the new filtered pixel V4. The filtered values of V4 and V5 replace the old values in the image to be displayed. The edge difference between pixels V4 and V5 is thus smoothed by D, reducing the change in pixel value from V4 to V5. This reduces visible color change at the block edge.

[0025] When the absolute value of the frequency component A0 (from the edge-crossing pixels) is greater than the quantization parameter QP, then condition checker 32 disables filtering for the current row or column. Large pixel differences may be caused by a real edge in the image, while smaller pixel differences are more likely caused by compression noise. When the absolute value of A0 is less than or equal to QP, then applicator 30 is enabled to add D to pixel V5 and subtract D from pixel V4.

[0026] A large amount of computational work is required for each pair of edge pixels that are smoothed. In particular, vector multipliers 11, 14, 16 each perform four multiplies and three adds, and a final divide-by-eight or 3-bit left-shift. While one arithmetic-logic-unit (ALU) or multiplier/adder/divider could be re-used three times, either the hardware required or the number of clock cycles to perform the three vector multiplies is significant. A total of 12 multiply operations is needed, and each multiply can be a full integer multiply rather than a simple right-shift multiply.

[0027] Since there are 300 or more blocks in each frame, and the de-blocking process may be repeated 16 times for each block, many computations may be performed by the de-blocking filter. It is therefore desirable to reduce computational complexity of the de-blocking filter. A more streamlined de-blocking filter is desirable.

BRIEF DESCRIPTION OF DRAWINGS

[0028] Figure 1 shows an image frame divided into rows and columns of blocks.

[0029] Figures 2A-B show prior-art de-blocking filter applied to vertical and horizontal block edges.

[0030] Figure 3 is a diagram highlighting computational steps performed during a prior-art de-blocking process.

[0031] Figure 4 highlights a computationally-simplified de-blocking process.

[0032] Figure 5 is a graph of pixel values near a block edge before and after smoothing.

DETAILED DESCRIPTION

[0033] The present invention relates to an improvement in de-blocking filters. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

[0034] The inventor has realized that much of the computational work of the prior-art de-blocking method is consumed by the vector multiplies. A large number of pixels are input early in the de-blocking process to generate frequency components for smoothing.

[0035] The inventor realizes that computational work can be reduced by inputting fewer pixels for smoothing. Rather than inputting a large number of pixels, such as 8 pixels, only the 2 edge pixels are used to generate smoothing terms. However, the result of smoothing is applied to a wider range of pixels than just the 2 edge pixels. The smoothing difference is applied in decreasing amounts to pixels farther away from the edge. This produces a more visually appealing edge, since pixel changes are gradually applied across more pixels than just the two edge pixels.

[0036] Figure 4 highlights a computationally-simplified de-blocking process. Only the edge pixels V4, V5 are applied to the early parts of the de-blocking process. The full range V1, V2..V8 of 9 pixels is not input until the end of the de-blocking process. The three vector multiplies of the prior art are eliminated, reducing computational load.

[0037] Edge pixels V4 of the current block and V5 of the next block are input to pixel differencer 48. Pixel differencer 48 generates the edge-pixel difference DE as (V4-V5)/2. The edge difference DE is input to minimum selector 40.

[0038] The quantization parameter QP for the current block N is input to maximum generator 42, which generates the maximum-allowed edge-pixel difference MD as 4 + (QP/2). This equation is empirically determined to give good results, but other equations such as 5 + QP/4 or 6+QP could be substituted. Minimum selector 40 selects the minimum of either MD or DE as its output. When the edge-pixel difference DE is large, greater than MD, then the quantization-parameter limited MD is selected by minimum selector 40. Otherwise edge-pixel difference DE is passed through.

[0039] The maximum-allowed edge-pixel difference MD is negated by negator 52, producing the two's complement of MD, -MD. Maximum selector 44 receives -MD and the minimum selected by minimum selector 40. The maximum is selected by maximum selector 44 and output as generated difference D. When edge-pixel difference DE is a negative number, it is passed through minimum selector

40, since QP is always positive. Then maximum selector 44 passes DE when DE is closer to 0 than -MD. Thus minimum selector 40 limits positive values of DE to within +MD, while maximum selector 44 limits negative values of DE to be between -MD and 0. A clipper could also be used.

[0040] The generated difference D from maximum selector 44 is then applied to divider/multiplier 54 to generate gradual smoothing values to apply to pixels. From generated difference D, positive and negative values of one-quarter, one-half, and three-quarters of D are generated, along with -D. Smoothing values -D/4, D/4, -D/2, D/2, -3D/4, 3D/4, and D are output by divider/multiplier 54.

[0041] Applicator 60 then applies the smoothing values generated by divider/multiplier 54 to a range of pixels in the current row or column. The full generated difference D is subtracted from edge pixel V4 in the current block, while three-quarters of D is added to edge pixel V5 in the next block.

[0042] Three-quarters of D is subtracted from adjacent pixel V3, while half of D is subtracted from next adjacent pixel V2 and only one-quarter of D is subtracted from more remote pixel V1.

[0043] In the next block, half of D is added to pixel V6, and one-quarter of D is added to pixel V7. Thus progressively smaller differences are added to pixels in the next block that are farther away from the block edge. Likewise, progressively smaller differences are subtracted from pixels in the current block that are farther away from the block edge.

[0044] Applicator 60 receives all 7 pixels and adjusts all 7 pixels to smooth the edge difference as follows in the current block:

[0045] $V1 = V1 - D/4$

[0046] $V2 = V2 - D/2$

[0047] $V3 = V3 - 3D/4$

[0048] $V4 = V4 - D$

[0049] In the next block, applicator 60 continues to smooth the edge difference:

[0050] $V5 = V5 + 3D/4$

[0051] $V6 = V6 + D/2$

[0052] $V7 = V7 + D/4$

[0053] Figure 5 is a graph of pixel values near a block edge before and after smoothing. The luminance Y values of pixels is plotted as the vertical axis while the pixel position along a row or column is shown on the horizontal axis. An abrupt change in pixel values occurs at the block boundary, between pixels V4 and V5. Compression approximations and losses that vary between the two blocks cause pixels V1, V2, V3, V4 in one block to have a higher Y value of Y2 than for pixels V5, V6, V7, V8 in the other block, which have pixel Y value Y1. In the real, uncompressed image, the Y pixel values decrease linearly from pixel V1 to V7, but compression causes the abrupt step in Y pixel values at the block edge.

[0054] The original, un-filtered pixel Y values are shown as X's in Fig. 5, while the new, filtered pixel Y values are shown as circles. As can be seen in Fig. 5, the un-filtered pixels (X's) show an abrupt step at the block boundary. However, the filtered pixels (O's) show a smooth transition between the two blocks.

[0055] The original pixel difference between pixels V4 and V5 at the block edge is Y2-Y1. The generated pixel difference D is half of this gap. Applicator 60 subtracts D from edge pixel V4 to get the filtered or smoothed V4 pixel, while 3/4 D is added to pixel V5. The new pixel difference at the block edge is Y2'-Y1', or about one-quarter of the original difference.

[0056] However, the pixel difference is not just smoothed between the two pixels V4, V5 on the block edge. Instead, progressively smaller amounts of the pixel difference D is added or subtracted from other pixels farther from the block edge. Pixels V2 and V6 have D/2 subtracted and added, while pixels V1 and V7 are adjusted by D/4. The result is a more gradual gradient of Y values across the block boundary.

[0057] ALTERNATE EMBODIMENTS

[0058] Several other embodiments are contemplated by the inventor. For example smoothing can be performed on a subset of the pixel components, such as just the luminance Y values, or on all pixel components, including the U, V chrominance values. The U, V components may be present for only half or one quarter of the pixel locations, such as for Bayer patterns and variants. RGB or other pixels could also be filtered.

[0059] All horizontal edges in a row of block, a pair of rows of blocks (one macroblock row), or in the entire image could be processed first, then the vertical edges processed. One macroblock of four 8x8 blocks or each 8x8 block could be processed for both horizontal and vertical before moving to the next macroblock of block. Other block sizes could be substituted, and different image or frame sizes could be processed.

[0060] The functional and computational blocks can be implemented in a variety of ways, such as by firmware routines in a digital-signal processor (DSP) chip, or in logic in a logic array chip, or as software routines executed by a processor, or a combination of techniques. The blocks can be partitioned in many different ways. A programmable register can allow calculations to be disabled, or allow for different threshold values or equations to be used to generated the maximum-allowed edge-pixel difference MD.

[0061] Other video formats, frame sizes, and block sizes could be supported. Many other functional blocks can exist in a complex MPEG decoder, and pipelining logic and staging registers may also be present. Various pipelining registers can be added. Different versions of the MPEG or other compression standards could be supported.

[0062] The abstract of the disclosure is provided to comply with the rules requiring an abstract, which will allow a searcher to quickly ascertain the subject matter of the technical disclosure of any patent issued from this disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. 37 C.F.R. § 1.72(b). Any advantages and benefits described may not apply to all embodiments of the invention. When the word "means" is recited in a claim element, Applicant intends for the claim element to fall under 35 USC § 112, paragraph 6. Often a label of one or more words precedes the word "means". The word or words preceding the word "means" is a label intended to ease referencing of claims elements and is not intended to convey a structural limitation. Such means-plus-function claims are intended to cover not only the structures described herein for performing the function and their structural equivalents, but also equivalent structures. For example, although a nail and a screw have different structures, they are equivalent structures since they both perform the function of fastening. Claims that do not use the word means are not intended to fall under 35 USC § 112, paragraph 6. Signals are typically electronic signals, but may be optical signals such as can be carried over a fiber optic line.

[0063] The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.